CLAIMS

1. A computer program product for representing and implementing a concept with a first functional domain and a second functional domain, wherein the first domain includes a first component representing the concept in a first functional domain, the first component having a first semantic usability in the first functional domain, and wherein the computer program product comprises:

a computer-readable medium having computer-readable signals stored thereon,

wherein the signals define a proxy component representing the concept in the second functional domain, the proxy component defined to cause execution of the first component in the first functional domain, and

wherein the proxy component has a second semantic usability in the second functional domain and the second semantic usability closely corresponds to the first semantic usability.

2. The computer program product of claim 1, wherein the proxy component was generated by a transformation performed on the first component.

3. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first mutability, and

wherein the second semantic usability of the proxy component is defined at least by a second mutability of the proxy component determined from at least the first mutability of the first component.

4. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first accessibility, and

wherein the second semantic usability of the proxy component is defined at least by a second accessibility of the proxy component determined from at least the first accessibility of the first component.

5. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first instantiability, and

wherein the second semantic usability of the proxy component is defined at least by a second instantiability of the proxy component determined from at least the first instantiability of the first component.

6. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first inheritability, and

wherein the second semantic usability of the proxy component is defined at least by a second inheritability of the proxy component determined from at least the first inheritability of the first component.

7. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first context usability, and

wherein the second semantic usability of the proxy component is defined at least by a context usability of the proxy component determined from at least the first context usability of the first component.

8. The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first polymorphicability, and

wherein the second semantic usability of the proxy component is defined at least by a polymorphicability of the proxy component determined from at least the first polymorphicability of the first component.

9. The computer program product of claim 1, wherein the first functional domain is the Java programming language and the second functional domain is the C++ programming language.

10. The computer program product of claim 9, wherein the first component is a first Java component and the proxy component is a C++ proxy component, and

wherein the C++ proxy component is aware of a context in which it is defined.

11. The computer program product of claim 9, wherein the first component is a first Java component and the proxy component is a C++ proxy component, and

wherein the C++ proxy component includes one or more proxy support elements.

12. The computer program product of claim 11, wherein the C++ proxy component is a C++ proxy class.

13. The computer program product of claim 12, wherein one or more proxy support elements of the C++ proxy class allow an instance of the C++ proxy class to be context-aware.

14. The computer program product of claim 13, wherein one or more of the proxy support elements that allow an instance of the C++ proxy class to be context-aware are context constructors.

15. The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy instance field.

16. The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy static field.

17. The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy array element.

18. The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy stand-alone object.

19. The computer program product of claim 13, wherein the C++ proxy class includes a proxy layer, and awareness of the contexts is required by the proxy layer.

20. The computer program product of claim 19, wherein the proxy layer is coded using the Java Native Interface.

21. The computer program product of claim 9, wherein the first component is a Java package and the proxy component is a C++ namespace.

22. The computer program product of claim 9, wherein the first component is a Java class and the proxy component is either a C++ proxy class or a C++ proxy struct.

23. The computer program product of claim 22, wherein the proxy component includes one or more proxy support elements.

24. The computer program product of claim 23, wherein one or more of the proxy support elements allow an instance of the C++ proxy class to be context-aware.

25. The computer program product of claim 22, wherein the Java class is declared abstract, and the C++ proxy component is instantiable.

26. The computer program product of claim 9, wherein the first component is a Java array type and the proxy component is either a C++ proxy class or a C++ proxy struct.

27. The computer program product of claim 26, wherein the proxy component includes one or more proxy support elements.

28. The computer program product of claim 27, wherein one or more of the proxy support elements allow an instance of the proxy component to be context-aware.

29. The computer program product of claim 26, wherein the proxy component includes a length field corresponding to a length attribute of the Java array.

30. The computer program product of claim 26, wherein the proxy component includes one or more array subscript operators that return a context-aware instance of the proxy array's element type.

31. The computer program product of claim 26, wherein the proxy component is a primitive array proxy component including one or more array subscript operators that return a primitive value.

32. The computer program product of claim 9, wherein the first component is a Java interface and the proxy component is either a C++ proxy class or a C++ proxy struct.

33. The computer program product of claim 32, wherein the first component includes one or more proxy support elements.

34. The computer program product of claim 33, wherein one or more of the proxy support elements allow an instance of the proxy component to be context-aware.

35. The computer program product of claim 32, wherein the proxy component is instantiable.

36. The computer program product of claim 9, wherein the first component is a Java method and the proxy component is a C++ proxy method.

37. The computer program product of claim 36, wherein the C++ proxy method has a constness determined from a mutability of the Java method.

38. The computer program product of claim 36, wherein the C++ proxy method declares a return type that has a mutability determined from a mutability of a return type declared by the Java method.

39. The computer program product of claim 36, wherein the C++ proxy method passes one or more arguments, each argument having a mutability determined from a mutability of a corresponding argument passed by the Java method.

40. The computer program product of claim 36, wherein the C++ proxy method is defined to throw an exception based on the Java method being defined to throw an exception.

41. The computer program product of claim 36, wherein the C++ proxy method is defined not to throw an exception based on the Java method being defined to throw an exception.

42. The computer program product of claim 36, wherein whether the C++ proxy method is declared virtual or not declared virtual is determined from one or more of the following aspects of the Java method: polymorphicability, mutability, and instantiability.

43. The computer program product of claim 9, wherein the first component is a Java field and the proxy component is a C++ proxy field.

44. The computer program product of claim 43, wherein the C++ proxy field is declared to be of type C++ proxy class, the C++ proxy class including one or more proxy support elements that allow an instance of the C++ proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware.

45. The computer program product of claim 43, wherein the Java field is of primitive type and the C++ proxy field is declared to be of type C++ primitive proxy class.

46. The computer program product of claim 45, wherein the C++ primitive proxy class includes one or more proxy support elements that allow an instance of the C++ primitive proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware.

47. The computer program product of claim 45, wherein one or more proxy support elements of the C++ primitive proxy is an overloaded operator that permits instances the C++ primitive proxy class 52 to be used on the left-hand side of one or more syntactical productions.

48. The computer program product of claim 43, wherein the C++ proxy field has a mutability determined from a mutability of the Java field.

49. A method of porting a digital entity from a first functional domain to a second functional domain, the digital entity including a first component that defines at least a first concept, the method comprising acts of:

(a) creating a second component in the second functional domain, the second component defining at least the first concept;

(b) transforming the second component into a first proxy component of the first functional domain; and

(c) replacing the first component in the digital representation with the first proxy component.

50. The method of claim 49, wherein the digital entity includes a plurality of components, the method further comprising the act of:

(d) for at least a third component of the digital entity, performing acts (a) - (c).

51. The method of claim 50, wherein the first and third components are related to a fourth component of the digital entity, the method further comprising an act of:

(e) after performing steps (a) - (c) on the first and third components, performing steps (a) - (c) on the fourth component to produce a proxy component of the fourth component.

52. The method of claim 49, further comprising acts of:

(e) executing the first proxy component of the digital entity in the first functional domain at runtime; and

(f) as a result of interpreting the first proxy component, executing the second component in the second functional domain.

53. The method of claim 49, wherein act (b) includes:

(i) generating a robust model from the second component; and

(ii) generating the first proxy component from the robust model.

54. The method of claim 49, wherein the first proxy component has a semantic usability in the first domain closely corresponding to the semantic usability of the second component in the second domain.

55. The method of claim 54, wherein the first proxy component includes one or more of proxy support elements.

56. The method of claim 55, wherein one or more of the proxy support elements allow an instance of the first proxy class to be context-aware.

57. A method of modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the method comprising acts of:

(a) receiving the first component; and

(b) generating a first model of the first component, including:

(i) for each subcomponent of the first component, generating a discrete element of the first model to represent the subcomponent; and

(ii) providing the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to:

determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute,

if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and

if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

58. The method of claim 57, wherein the first model is operative to be transformed into a second component of a second domain that defines at least the concept defined by the first component.

59. The method of claim 57, wherein the first model is operative to be transformed into a proxy component of a second domain that wraps the first component.

60. The method of claim 57, wherein the first component is a source component operative to be reproduced from the first model.

61. The method of claim 57, wherein the first component is received in parsed form.

62. The method of claim 57, wherein each discrete element is queryable as to its attributes.

63. The method of claim 57, wherein each discrete element is modifiable.

64. The method of claim 57, wherein act (b)(ii) includes:

maintaining relationship information regarding a relationship between the first model and the first component.

65. The method of claim 57, wherein act (b)(ii) includes:

for one or more of the discrete elements, maintaining relationship information regarding a relationship between the discrete element and a subcomponent that it represents.

66. The method of claim 57, wherein the concept is a dynamic concept.

67. The method of claim 57, further comprising an act of:

(c) adding additional information to the first model.

68. The method of claim 57, further comprising an act of:

(c) editing the first model.

69. The method of claim 57, further comprising an act of:

(c) transforming the first model into a second model of a second component of a second domain that defines at least the concept defined by the first component.

70. The method of claim 69, wherein the second model is operative to be transformed into the second component of the second domain.

71. The method of claim 69, wherein the second model is operative to be transformed into a proxy component of a second domain that wraps the first component.

72. The method of claim 69, wherein the second component includes one or more subcomponents, and act (c) includes:

(i) for each subcomponent of the second component, generating a discrete element of the second model to represent the subcomponent; and

(ii) providing the second model with a property of relationship awareness such that, if a second discrete element or attribute of the second model is changed, the second model is operative to:

determine if the second discrete element or attribute has one or more elements and attributes related to the second discrete element or attribute,

if the second discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and

if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed second discrete element or attribute.

73. The method of claim 72, wherein act (c)(ii) includes:

maintaining relationship information regarding a relationship between the second model and the second component.

74. The method of claim 72, wherein act (c) (ii) includes:

for one or more of the discrete elements of the second model, maintaining relationship information regarding a relationship between the discrete element and a subcomponent of the second component that it represents.

75. The method of claim 9, wherein the second domain is a functional domain.

76. A system for modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the system comprising:

means for receiving the first component; and

means for generating a first model of the first component, including:

means for generating, for each subcomponent of the first component, a discrete element of the first model to represent the subcomponent; and

means for providing the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to:

determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute,

if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and

if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

77. A system for modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the system comprising:

a model generator to receive the first component, to generate a first model including:

for each subcomponent of the first component, a discrete element of the first model representing the subcomponent; and

one or more model concepts that provide the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to:

determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute,

if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and

if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

78. A computer program product for modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the system comprising:

a computer-readable medium having computer-readable signals stored thereon,

wherein the signals define a model generator to receive the first component, to generate a first model including:

for each subcomponent of the first component, a discrete element of the first model representing the subcomponent; and

one or more model concepts that provide the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to:

determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute,

if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and

if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

79. A method of transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the method comprising acts of:

(a) determining a type of the first component; and

(b) transforming the first component into the proxy component in accordance with the determined type,

wherein the proxy component defines at least the concept defined by the first component.

80. The method of claim 79, wherein the first component is in parsed form.

81. The method of claim 79, wherein the first component is a source component.

82. The method of claim 79, wherein the first component is a compiled component.

83. The method of claim 79, wherein act (b) includes:

(i) generating a model from the first component; and

(ii) generating the proxy component from the model.

84. The method of claim 79, wherein the proxy component has a semantic usability in the second domain closely corresponding to the semantic usability of the first component in the first domain.

85. The method of claim 84, wherein the proxy component includes one or more proxy support elements to allow an instance of the proxy component to be context-aware.

86. The method of claim 79, wherein the first domain is a first programming language.

87. The method of claim 86, wherein the first programming language is a high-level programming language.

88. The method of claim 87, wherein the first programming language is Java.

89. The method of claim 88, wherein the second domain is C++.

SUB A3 90. The method of claim 89, wherein the first component is one of the following Java components: a Java interface, and a Java class, and act (b) includes:

(i) transforming the Java component into a C++ proxy class.

91. The method of claim 90, wherein the C++ proxy class includes one or more proxy support elements.

92. The method of claim 89, wherein the first component is a Java array and act (b) includes:

(i) transforming the Java array into a C++ proxy class.

93. The method of claim 92, wherein the C++ proxy class includes one or more proxy support elements.

94. The method of claim 89, wherein the first component is a Java method and act (b) includes:

(i) transforming the Java method into a C++ proxy method.

95. The method of claim 89, wherein the first component is a Java field and act (b) includes:

(i) transforming the Java field into a C++ proxy field.

96. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first mutability, and act (b) includes:

(i) determining a second mutability of the proxy component from at least the first mutability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second mutability.

97. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first accessibility, and act (b) includes:

(i) determining a second accessibility of the proxy component from at least the first accessibility of the first component, wherein the second semantic usability of the proxy component is defined at least by the second accessibility.

98. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first instantiability, and act (b) includes:

(i) determining a second instantiability of the proxy component from at least the first instantiability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second instantiability.

99. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first inheritability, and act (b) includes:

(i) determining a second inheritability of the proxy component from at least the first inheritability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second instantiability.

100. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first context usability, and act (b) includes:

(i) determining a second context usability of the proxy component from at least the first context usability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second context usability.

101. The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first polymorphicability, and act (b) includes:

(i) determining a second polymorphicability of the proxy component from at least the first polymorphicability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second polymorphicability.

102. A system for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the system comprising:

means for determining a type of the first component; and

means for transforming the first component into the proxy component in accordance with the determined type, wherein the proxy component defines at least the concept defined by the first component.

103. A system for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the system comprising:

a component transformer to receive as input the first component, to determine a type of the first component, to transform the first component into the proxy component in accordance with the determined type, and to output the proxy component,

wherein the proxy component defines at least the concept defined by the first component.

104. A computer program product for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the computer program product comprising:

a computer-readable medium having computer-readable signals thereon,

wherein the signals define a component transformer to receive as input the first component, to determine a type of the first component, and to transform the first component into the proxy component in accordance with the determined type,

wherein the proxy component defines at least the concept defined by the first component.

ADD A4